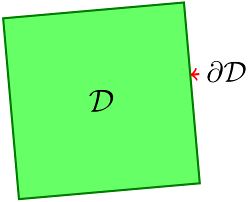
WaveToy Thorn: Wave Equation

For a given source function $S(x, y, z, t)$ find a scalar wave field $\varphi(x, y, z, t)$ inside the domain $\mathcal{D}$ with a boundary condition:



- inside $\mathcal{D}$:

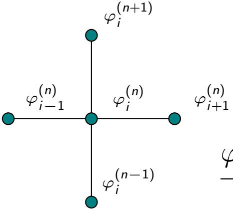
$$\frac{\partial^2\varphi}{\partial t^2} = c^2\Delta\varphi + S$$

- on the boundary $\partial\mathcal{D}$:

$$\varphi_{|\partial\mathcal{D}} = 0$$

WaveToy Thorn: Discretization

Discretization:
approximating continuous function $\varphi(x, y, t)$ with a grid function $\varphi^{(n)}_{i,j}$:



$$\frac{\partial^2\varphi}{\partial t^2} = c^2(\partial_x^2\varphi + \partial_y^2\varphi) + S$$

$$\Downarrow (c \equiv 1)$$

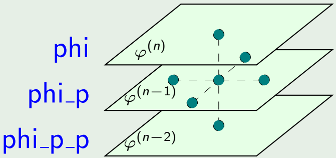
$$\frac{\varphi^{(n+1)}_{i,j} - 2\varphi^{(n)}_{i,j} + \varphi^{(n-1)}_{i,j}}{2\Delta t^2} = \frac{\varphi^{(n)}_{i+1,j} - 2\varphi^{(n)}_{i,j} + \varphi^{(n)}_{i-1,j}}{2\Delta x^2} + \frac{\varphi^{(n)}_{i,j+1} - 2\varphi^{(n)}_{i,j} + \varphi^{(n)}_{i,j-1}}{2\Delta y^2} + S^{(n)}_{i,j}$$

WaveToy Thorn

Thorn structure:

interface.ccl

- grid function phi[3]:



- Boundary_SelectVarForBC

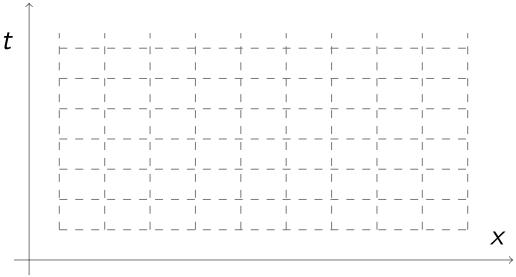
param.ccl

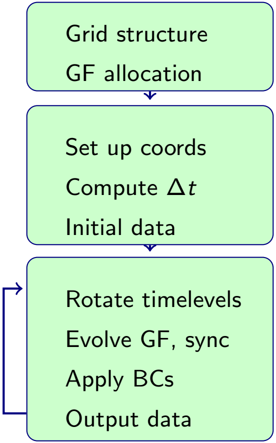
- Parameters of initial Gaussian pulse: amplitude A, radius R, width σ

schedule.ccl

- WaveToy_InitialData
- WaveToy_Evolution
- WaveToy_Boundaries

WaveToy Thorn: Algorithm Illustration





Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

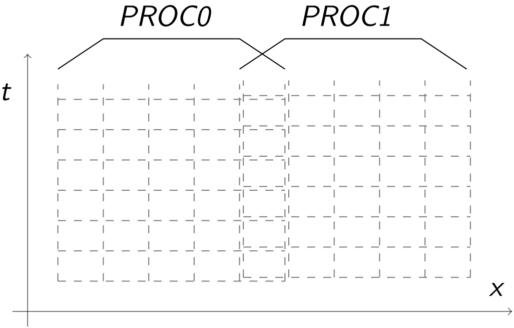
Rotate timelevels

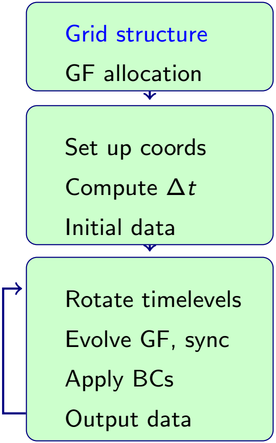
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration




Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

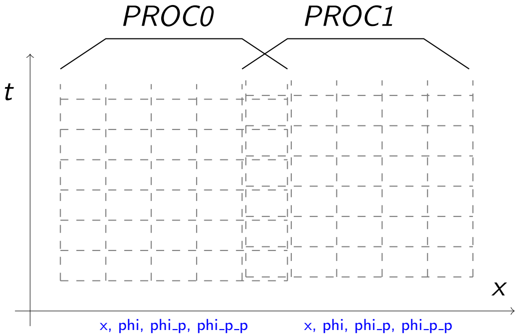
Rotate timelevels

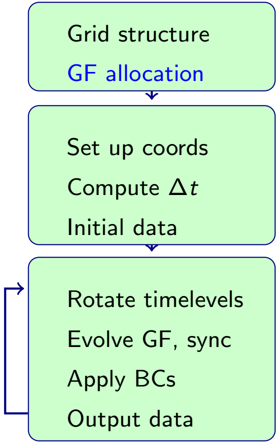
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration





Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

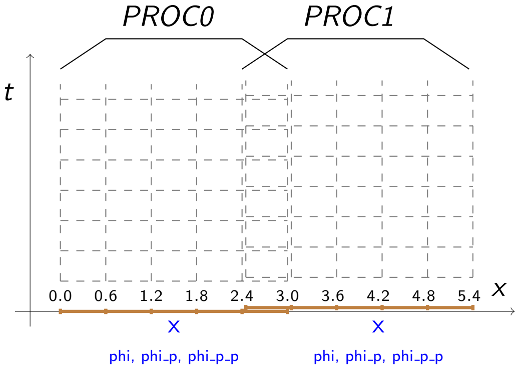
Rotate timelevels

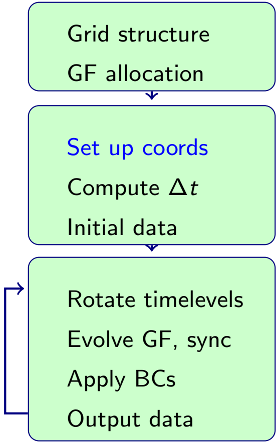
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration





Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

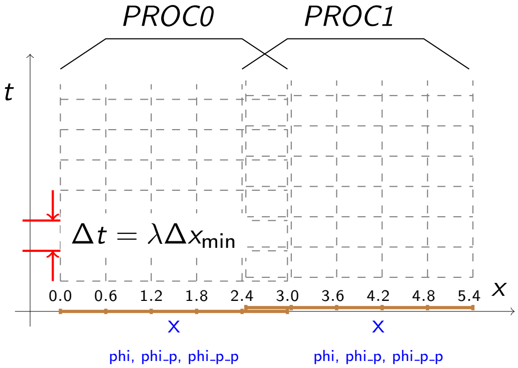
Rotate timelevels

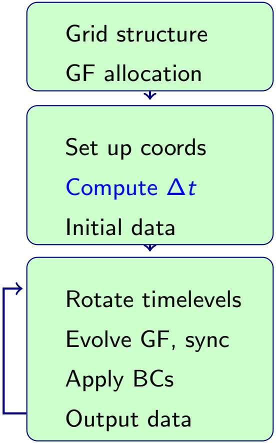
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration





Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

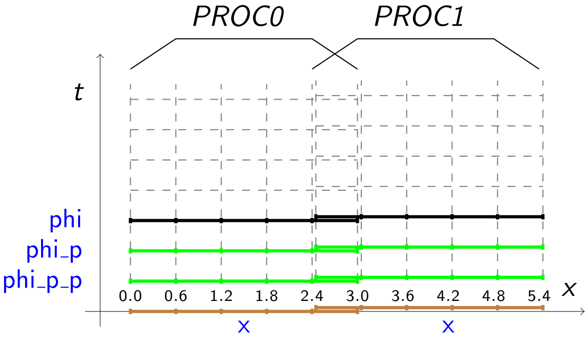
Rotate timelevels

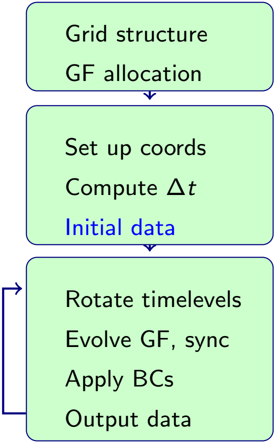
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration





Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

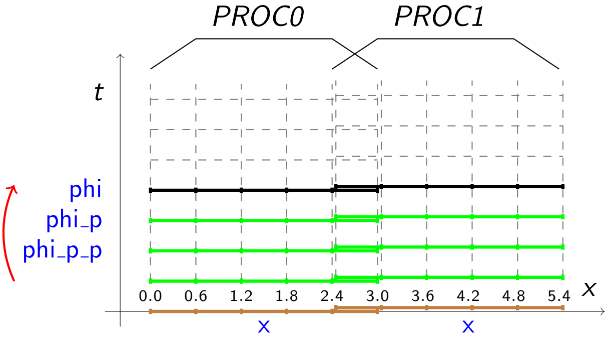
Rotate timelevels

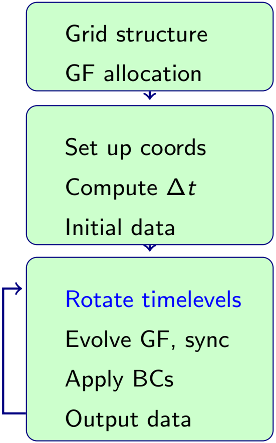
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration




Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

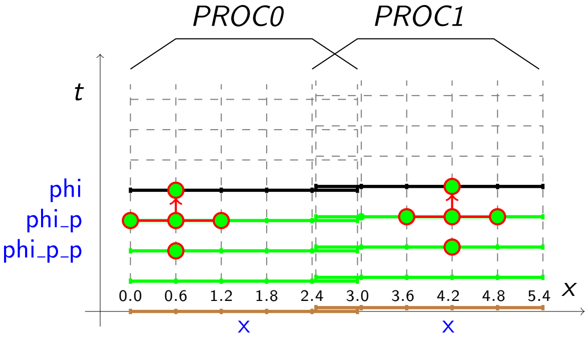
Rotate timelevels

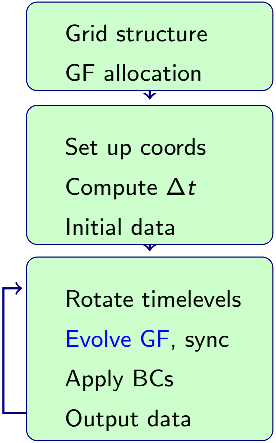
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration




Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

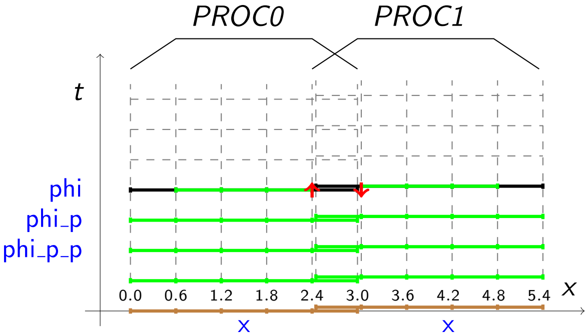
Rotate timelevels

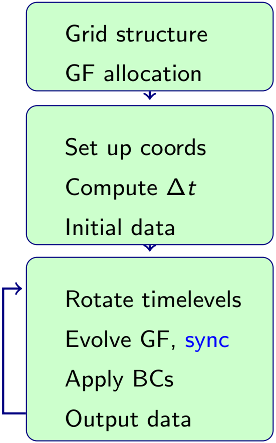
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration





Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

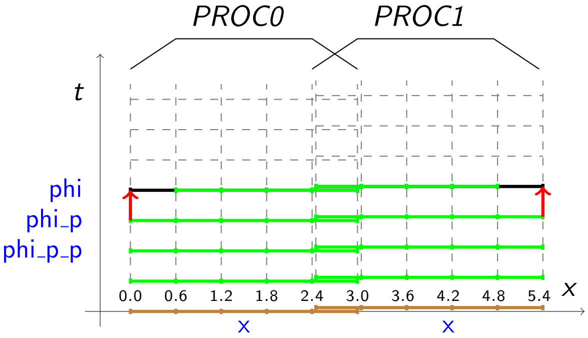
Rotate timelevels

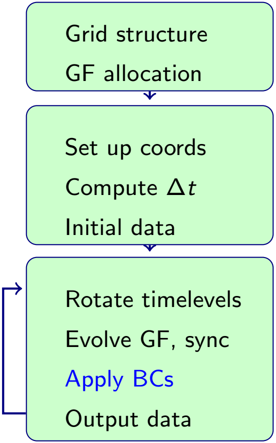
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration





Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

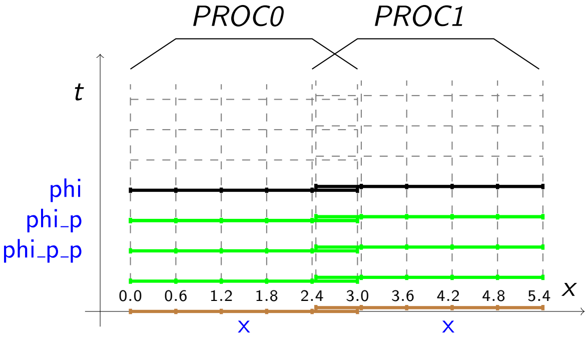
Rotate timelevels

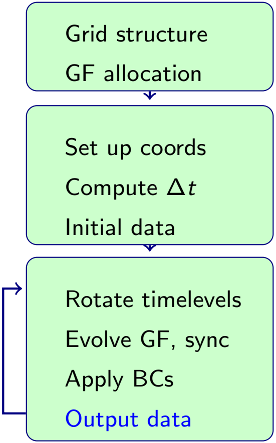
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration





Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

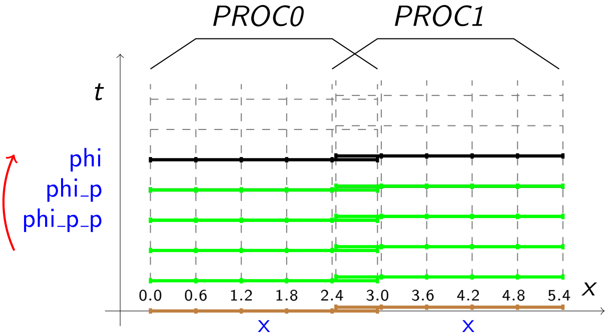
Rotate timelevels

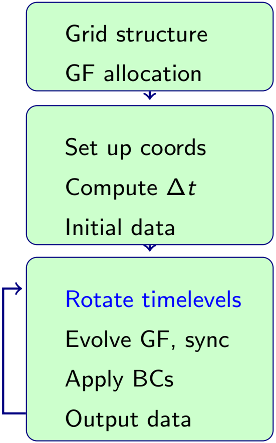
Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn: Algorithm Illustration





Grid structure

GF allocation

Set up coords

Compute Δt

Initial data

Rotate timelevels

Evolve GF, sync

Apply BCs

Output data

WaveToy Thorn

Directory structure:

```
WaveToy/
|-- COPYRIGHT
|-- README
|-- configuration.ccl
|-- doc
|   `-- documentation.tex
|-- interface.ccl
|-- schedule.ccl
|-- param.ccl
`-- src
    |-- WaveToy.c
    `-- make.code.defn
```

WaveToy Thorn

Directory structure:

```
WaveToy/
|-- COPYRIGHT
|-- README
|-- configuration.ccl
|-- doc
|   `-- documentation.tex
|-- interface.ccl
|-- schedule.ccl
|-- param.ccl
`-- src
    |-- WaveToy.c
    `-- make.code.defn
```

WaveToy Thorn

- `interface.ccl`:

```
IMPLEMENTS: wavetoy_simple
INHERITS: grid

PUBLIC:

CCTK_REAL scalarevolve TYPE=gf TIMELEVELS=3
{
  phi
} "The evolved scalar field"

CCTK_INT FUNCTION Boundary_SelectVarForBC( \
  CCTK_POINTER_TO_CONST IN GH, CCTK_INT IN faces, \
  CCTK_INT IN boundary_width, CCTK_INT IN table_handle, \
  CCTK_STRING IN var_name, CCTK_STRING IN bc_name)

REQUIRES FUNCTION Boundary_SelectVarForBC
```

WaveToy Thorn cont.

- `schedule.ccl`:

```
STORAGE: scalarevolve[3]

SCHEDULE WaveToy_InitialData AT CCTK_INITIAL
{
  LANG: C
} "Initial data for 3D wave equation"

SCHEDULE WaveToy_Evolution AT CCTK_EVOL
{
  LANG: C
  SYNC: scalarevolve
} "Evolution of 3D wave equation"

SCHEDULE WaveToy_Boundaries AT CCTK_EVOL AFTER WaveToy_Evolution
{
  LANG: C
} "Select boundary conditions for the evolved scalar"

SCHEDULE GROUP ApplyBCs as WaveToy_ApplyBCs AT CCTK_EVOL AFTER WaveToy_Boundaries
{
} "Apply boundary conditions"
```

WaveToy Thorn cont.

- param.ccl:

```
CCTK_REAL amplitude "The amplitude of the waves"
{
 *:* :: "Anything"
} 1.0

CCTK_REAL radius "The radius of the gaussian wave"
{
 0:* :: "Positive"
} 0.0

CCTK_REAL sigma "The sigma for the gaussian wave"
{
 0:* :: "Positive"
} 0.1
```

WaveToy Thorn cont.

Directory structure:

```
WaveToy/
|-- COPYRIGHT
|-- README
|-- configuration.ccl
|-- doc
|   `-- documentation.tex
|-- interface.ccl
|-- schedule.ccl
|-- param.ccl
`-- src
    |-- WaveToy.c
    `-- make.code.defn
```

WaveToy Thorn cont.

- WaveToy.c:

```
void  WaveToy_Evolution(CCTK_ARGUMENTS) {
  DECLARE_CCTK_ARGUMENTS;

  CCTK_REAL dt2,dx2i,dy2i,dz2i;

  dt2 = CCTK_DELTA_TIME*CCTK_DELTA_SPACE(2);
  dx2i = 1.0/(CCTK_DELTA_SPACE(0)*CCTK_DELTA_SPACE(0));
  dy2i = 1.0/(CCTK_DELTA_SPACE(1)*CCTK_DELTA_SPACE(1));
  dz2i = 1.0/(CCTK_DELTA_SPACE(2)*CCTK_DELTA_SPACE(2));

  /* Do the evolution */
  for (int k=1; k<cctk_lsh[2]-1; k++) {
    for (int j=1; j<cctk_lsh[1]-1; j++) {
      for (int i=1; i<cctk_lsh[0]-1; i++) {
        phi[CCTK_GFINDEX3D(cctkGH,i,j,k)] = 2*(1 - (dt2)*(dx2i + dy2i + dz2i)) *
                      phi_p[CCTK_GFINDEX3D(cctkGH,i,j,k)] - phi_p_p[CCTK_GFINDEX3D(cctkGH,i,j,k)]
                      + (dt2) *
                    ( ( phi_p[CCTK_GFINDEX3D(cctkGH,i+1,j  ,k  )]
                       +phi_p[CCTK_GFINDEX3D(cctkGH,i-1,j  ,k  )] )*dx2i
                     +( phi_p[CCTK_GFINDEX3D(cctkGH,i  ,j+1,k  )]
                       +phi_p[CCTK_GFINDEX3D(cctkGH,i  ,j-1,k  )] )*dy2i
                     +( phi_p[CCTK_GFINDEX3D(cctkGH,i  ,j  ,k+1)]
                       +phi_p[CCTK_GFINDEX3D(cctkGH,i  ,j,  k-1)] )*dz2i);
      }
    }
  }
}
```

WaveToy Thorn cont.

- WaveToy.c cont.:

```
void WaveToy_InitialData(CCTK_ARGUMENTS) {
  DECLARE_CCTK_ARGUMENTS;
  DECLARE_CCTK_PARAMETERS;

  for(int k=0; k<cctk_lsh[2]; k++) {
    for(int j=0; j<cctk_lsh[1]; j++) {
      for(int i=0; i<cctk_lsh[0]; i++) {
        int vindex =  CCTK_GFINDEX3D(cctkGH,i,j,k);
        CCTK_REAL dt = CCTK_DELTA_TIME;
        CCTK_REAL X = x[vindex];
        CCTK_REAL Y = y[vindex];
        CCTK_REAL Z = z[vindex];
        CCTK_REAL R = sqrt(X*X + Y*Y + Z*Z);

        phi[vindex] = amplitude*exp( - SQ( (R - radius) / sigma ) );

        if (R == 0.0) {
          phi_p[vindex] = amplitude*(1.0 - 2.0*dt*dt/sigma)*exp(-dt*dt/sigma);
        } else {
          phi_p[vindex] = amplitude/2.0*(R-dt)/R*
            exp( - SQ( (R - radius - dt)/ sigma ) )
            + amplitude/2.0*(R+dt)/R*
            exp( - SQ( (R - radius + dt)/ sigma ) );
        }
      }
    }
  }
}
```

WaveToy Thorn cont.

- `WaveToy.c`:

```
void WaveToy_Boundaries(CCTK_ARGUMENTS) {
  DECLARE_CCTK_ARGUMENTS;

  /* Uses all default arguments, so invalid table handle -1 can be passed */
  if (Boundary_SelectVarForBC (cctkGH, CCTK_ALL_FACES, 1, -1,
                                       "wavetoy_simple::phi", "scalar") < 0)  {
    CCTK_WARN (0, "WaveToy_Boundaries: Error selecting boundary condition");
  }
}
```

- `make.config.defn`:

```
SRCS = WaveToy.c
```

Parameter file

- Example parameter file:

```
Cactus::cctk_run_title = "Simple WaveToy"

ActiveThorns = "Time Boundary Carpet CarpetLib CartGrid3D CoordBase IOUtil
                CarpetIOBasic CarpetIOASCII CarpetIOHDF5 SymBase WaveToy"

Cactus::cctk_itlast = 10000
Time::dtfac = 0.5

IO::out_dir             = $parfile
IOBasic::outInfo_every  = 1
IOASCII::out1D_every    = 10
IOASCII::out1D_vars     = "wavetoy_simple::phi"

IOHdf5::out_every = 100
IOHdf5::out_vars        = "wavetoy_simple::phi"
```